REPRINT

Covert Channels — Here to Stay?

Ira S. Moskowitz and *Myong H. Kang*

FROM:
Proceedings of COMPASS '94, Gaithersburg, MD, June 27 - July 1, 1994, pages 235-243, IEEE Press.

CONTACT:
Ira S. Moskowitz, Information Technology Division, Mail Code 5543, Naval Research Laboratory, Washington, DC 20375.

Myong H. Kang, Information Technology Division, Mail Code 5542, Naval Research Laboratory, Washington, DC 20375.

E-MAIL:
moskowit@itd.nrl.navy.mil
mkang@itd.nrl.navy.mil

COMMENTS:(version — July, 7 1994)
As of June 24, 1994 this preprint has corrected two typographical errors that appeared in the actual published paper.
The spelling of atomicity on the first page, column 2 has been fixed. On the fourth page, column 1, middle of the page X has been changed to $H(X)$ when discussing the conditional entropy of independent random variables.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **1994**	2. REPORT TYPE	3. DATES COVERED **00-00-1994 to 00-00-1994**

4. TITLE AND SUBTITLE **Covert Channels - Here to Stay?**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**10**	

Covert Channels — Here to Stay?

Ira S. Moskowitz and Myong H. Kang
Information Technology Division
Naval Research Laboratory
Washington, DC 20375

Abstract

We discuss the difficulties of satisfying high-assurance system requirements without sacrificing system capabilities. To alleviate this problem, we show how trade-offs can be made to reduce the threat of covert channels.

We also clarify certain concepts in the theory of covert channels. Traditionally, a covert channel's vulnerability was measured by the capacity. We show why a capacity analysis alone is not sufficient to evaluate the vulnerability and introduce a new metric referred to as the "small message criterion".

1 Introduction

In this paper we discuss how covert channels arise in the area of high-assurance systems. We give an overview of covert channel theory, with examples, and advance our hypothesis that covert channels can never be totally eliminated in many "practical" high-assurance systems. A high-assurance system should perform the intended tasks of reliability, security, and performance as efficiently as possible, conflicts between the requirements are inherent.

The paper is organized as follows:

- We show how reliability and performance requirements can undermine efforts at thwarting covert channels.
- We look at covert channels in terms of information theory and clarify certain concepts.
- We suggest that a capacity analysis alone does not suffice when dealing with covert channels and introduce a new metric referred to as the "small message criterion".
- We discuss trade-offs between covert channel degradation and performance.
- We then discuss our recent work on the "pump" and show how it reduces the covert channel threat without degrading performance.

2 Practical High-Assurance Multilevel Systems

All multilevel systems require information flow from Low to High[1]. Two methods of information flow that do not violate BLP [1] are *read-down* and *blind write-up*. However, these methods have practical problems in terms of reliability and performance [11].

As the computing environment becomes more sophisticated, complicated operations are needed and other features, e.g., atomicity, become crucial requirements. One type of high-assurance system is a secure database system (DBS). Reliability and atomicity[2] of transactions in a secure DBS are integral components of high-assurance computing. In the following, we show how difficult it is to eliminate totally covert channels in today's sophisticated high-assurance computer systems.

Mathur and Keefe [13] showed that conflicts exist between atomicity and security in the case of multilevel transaction execution [4]. In other words, there may be no concurrency controller that can schedule multilevel transactions, and guarantee the atomicity of transactions and security simultaneously.

Let us consider the potential conflicts between reliability and security. In a DBS, a user/process wishes to receive an acknowledgement of a successful update. Without acknowledgements, necessary data may be written over, or may be lost during a crash, which is unacceptable in a high-assurance DBS. In a non-secure DBS there is no problem with acknowledgements; however in a secure DBS acknowledgements can allow a covert channel to exist between High and Low.

Consider the following example of an object-oriented DB program involving three objects: (1) EMPLOYEE, (2) PAY_INFO, and (3) WORK_INFO.

[1] If a multilevel system does not require information flow from Low to High then we consider it as two system-high systems.

[2] All or nothing without the appearance of interruptions.

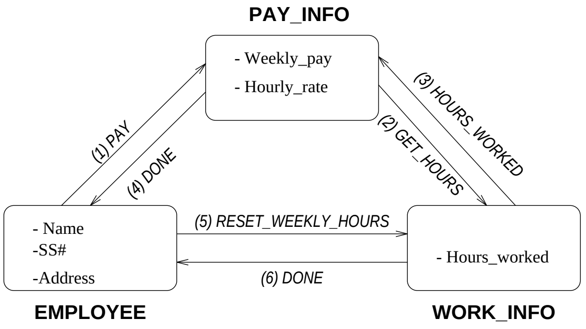

Figure 1: Objects in payroll database.

The main program calls the `employee` method in EMPLOYEE object (in C++ look-alike pseudocode):

```
employee()
{
    PAY_INFO.pay();
    WORK_INFO.reset_weekly_hours();
}
```

The `pay` method in turn:

```
pay()
{
    WORK_INFO.get_hours();
}
```

In conventional programming sense (i.e., if there is no parallelism between `PAY_INFO.pay()` and `WORK_INFO.reset_weekly_hours()`), the correctness of the program is guaranteed by executing `WORK_INFO.reset_weekly_hours()` after `PAY_INFO.pay()` is executed (see figure 1).

If the same transaction is performed in a secure system, where PAY_INFO is a high object and EMPLOYEE and WORK_INFO are low objects, then the above solution is not acceptable because the acknowledgement (4) from PAY_INFO (High) to EMPLOYEE (Low) can be used as a covert timing channel by PAY_INFO moderating the time at which the acknowledgement (4) is sent to EMPLOYEE.

To overcome this security problem, Jajodia and Kogan [10] proposed a message filter that enforces the security policy in multilevel object-oriented systems. Sandhu, Thomas, and Jajodia [21] proposed a covert channel free implementation strategy of this message filter in the kernelized architecture [7]. The proposed covert channel free solution is as follows (the heavy blocks in the diagram represent the message filters):

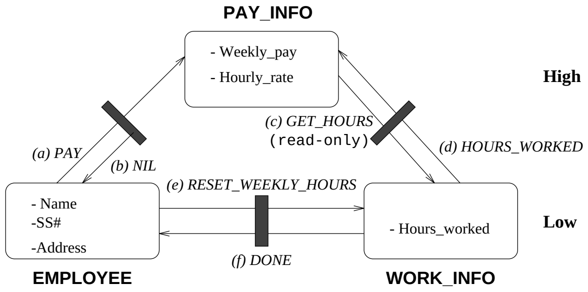

Figure 2: Proposed solution for a secure payroll database.

- The transaction is initiated by EMPLOYEE by sending the *PAY* message to PAY_INFO. If PAY_INFO can send an acknowledgement back to EMPLOYEE then a potential covert timing channel exists. Hence, the message filter sends *NIL* right away and blocks any response from High to Low.
- PAY_INFO, in turn, sends *GET_HOURS* to WORK_INFO to read `Hours_worked`. WORK_INFO should not know *when* or *by whom* its information is read[3] (if it knows then this information can be used as a covert channel).
- In the meantime, EMPLOYEE sends *RESET_WEEKLY_HOURS* to WORK_INFO to reset `Hours_worked`. WORK_INFO can send *DONE* to EMPLOYEE because they are at the same level.
- To guarantee that *GET_HOURS* reads the value of `Hours_worked` before *RESET_WEEKLY_HOURS* is executed, WORK_INFO uses a multiple version scheme. In other words, WORK_INFO always makes a new version whenever its information is updated so that High can read appropriate (but potentially old) versions of `Hours_worked`.

Even though the above solution is covert channel free, it has a few practical problems. Let us consider these problems:

- Since the computer resources are limited, some versions have to be deleted from the system after some time has passed (i.e., garbage collection). Therefore we cannot keep all old versions of `Hours_worked` and PAY_INFO might not read the correct version of `Hours_worked`.
- When *PAY* is sent to PAY_INFO, the message filter sends *NIL* right away. Hence, EMPLOYEE

[3] There is some controversy over how High can send `read-only` messages to Low without revealing its activity. However, in this paper, we assume that there is a covert channel free implementation to send read-only messages to Low.

cannot confirm whether PAY_INFO actually received a message or not—it just hopes that the message arrived safely at the destination. If, somehow, PAY_INFO is not ready to receive a message or the message passing system has some problems then the message may never get to PAY_INFO, and EMPLOYEE would never know what happened to the message. If EMPLOYEE knew that the message was not delivered to PAY_INFO, e.g., by not receiving the DONE message as in the non-secure version, it could repeat the message or abandon the task without resetting `Hours_worked`.

- Even if the message eventually gets to PAY_INFO, the correct version of `Hours_worked` may be deleted already because WORK_INFO does not know when, or by whom, the data is read.

We have shown that it is difficult to obtain reliability and atomicity in conjunction with security. Information theory can quantify some of these difficulties involving covert channels.

3 Information Theory Background

In brief, a *covert channel* is a communication channel that exists, contrary to design, in a computer system. There are three historical patterns of covert channel analysis that concern us. The first is the use of the term *bandwidth* [6] instead of capacity, the second is ignoring the encoding/decoding process, and the third is using capacity as the total measure of insecurity, instead of including factors such as the length of the message or the quality of the message. We will not concentrate on the second issue in this paper.

Covert channel analysis is just a subset of information theory. Information theory is concerned with sending signals from a transmitter to a receiver, with the possibility of noise degrading the signal fidelity. This process of transmission is a communication channel or simply a channel. In general, the transmitter takes a message and encodes it before it transmits it. The receiver, once it receives the message, decodes the message. The brilliance of Shannon's work is that it gives an upper limit on the rate at which messages can be passed, within a certain given error tolerance, through the communication channel — by the process of encoding, transmitting, receiving, and decoding — based solely on how noise affects the transmission of the signals. This upper limit is referred to as the *capacity* of the channel.

The inputs from the transmitter to the channel constitute the input alphabet. Sending an input letter across the channel is synonymous with sending a symbol across the channel. The interpretation of the received symbol by the receiver, prior to any decoding, is what constitutes the output alphabet. If both the input and output alphabets are discrete we have a *discrete channel*, which is usually the case in covert channel analysis. The choice of alphabets often approximates the actual physical process. This is especially true when the output alphabet is made up of time values, e.g., [18]. In general, the alphabets need not have anything in common. However, if no noise exists in the channel then what the transmitter puts in is what the receiver gets out, and thus the alphabets are identical.

For the sake of convenience, the standard unit of time is a tick (t). Capacity may be measured in both units of bits per channel usage (C_u) or in units of bits per tick (C_t). If every symbol takes the same amount of time τ to be sent across the communication channel (*constant time channel*), then $C_t = \tau^{-1} C_u$. Therefore, without loss of generality, we may use either measurement of capacity for a constant time channel. In the literature, the meaning is usually made clear by examining the units in which capacity is expressed.

A *storage channel* is a covert channel where the output alphabet consists of different responses all taking the same time to be transmitted. A *timing channel* is a covert channel where the output alphabet is made up of different time values corresponding to the same response. A *mixed channel*[4] is a combination of the two. Even though our definition of storage channel is not *de jure* identical to Lampson's [12], it is *de facto* the same. Our definitions capture the operational differences between storage, timing, and mixed channels.

For example, one type of of storage channel is given by Low requesting a resource and receiving the (constant time) reply that the resource can be used or that the resource cannot be used. An output alphabet of two distinct symbols results. In contrast, a timing channel is one where Low always receives a response that the resource is available for its use but receives the response at different times. Section 2 was concerned with a timing channel. In section 5 we discuss a storage channel.

For a timing (or mixed) channel, the capacities C_t and C_u no longer differ by a constant multiple. Let us first show how to calculate C_u. We will work with channels that are memoryless, unless otherwise noted. By *memoryless* we mean that there are no restrictions on what symbol may be transmitted based upon the prior history of the channel. Each transmission is independent of the past transmissions and they are not time-varying. We will mostly look at channels that are both discrete and memoryless (DMC). This is not a serious constraint because the capacity of a channel with memory can often be bound from above by a memoryless version of the channel [11].

For a random variable X, $X = x_i$, let $H(X)$ denote the *entropy* of X. The entropy measures the "information" or "surprise" of the different values of X. For a particular value x the surprise is $-\log P(x)$; if x_i happens with certainty then its surprise is zero, and if x_i never occurs then its surprise is maximal at infinity. We always use the base two logarithm so that the units of information are in bits. The entropy is the expected value of the information of X and has units of bits per

[4] This is our own terminology and we will not concentrate on these types of covert channels in this paper.

outcome. For the ease of notation we will express the event $(X = x_i)$ as (x_i). Thus

$$H(X) = -\sum_i P(x_i) \log P(x_i) \ .$$

If X is the random variable representing the input to a channel the unit *outcome* of X is synonymous with the unit *channel usage*.

Information theory is concerned with how the input or transmission entropy changes while it travels through the channel. If the channel is noiseless then the amount of information in a transmission should be unchanged. However, if there is noise in the channel then the fidelity of the signal is degraded and the information sent is diminished. If the channel noise is so great and all encompassing then there is no more surprise in seeing any one symbol over another. This is mathematically modeled by the equivocation or *conditional entropy* $H(X \mid Y)$, where X is the random variable representing the channel input and Y is random variable representing the channel output. Mathematically

$$\begin{aligned} H(X \mid Y) &= \sum_j H(X \mid y_j) P(y_j) \\ &= -\sum_{i,j} P(y_j) P(x_i \mid y_j) \log P(x_i \mid y_j) \ . \end{aligned}$$

A good way to understand $H(X \mid Y)$ is to look at the extreme cases; $H(X \mid X) = 0$ and $H(X \mid Y) = H(X)$ if X and Y are independent.

For a DMC the noise is expressed by the conditional probabilities $p_{ij} = P(y_j \mid x_i)$. Thus we see how the distributions for X and the noise totally determine both $H(X)$ and $H(X \mid Y)$. The mutual information in units of bits per channel usage $I_u(X, Y)$ measures how much information is actually sent across the channel from input X to receiver Y. The mutual information in units of bits per channel usage is the difference between the input entropy and the conditional entropy:

$$I_u(X, Y) = H(X) - H(X \mid Y) \ .$$

When transmitting, the transmitter can do nothing about the noise, and the receiver is passive and waits for symbols to be passed over the channel. However, the transmitter can send different symbols with different frequencies; thus, there are different distributions for X. By changing the frequency of the symbols sent, the transmitter can affect the amount of information sent to the receiver. C_u is the maximum amount of information, in units of bits per channel usage, that can be sent over the DMC:

$$C_u = \max I_u(X, Y) \ ,$$

where the maximum is taken over the different distributions on X.

Let t_i be the amount of time to transmit input letter x_i across the DMC. The random variable T is the time to send a symbol and the distribution of T is determined by the distribution of X. The mean of T, in units of ticks per channel usage, is represented by $E(T)$. The mutual information in units of bits per tick $I_t(X, Y)$ for a DMC is

$$I_t(X, Y) = \frac{I_u(X, Y)}{E(T)} \ .$$

The capacity in units of bits per tick for a DMC is given by [23]

$$C_t = \max \frac{I_u(X, Y)}{E(T)} \ , \qquad (1)$$

maximized as before. Of course, if this is a constant time DMC the value $E(T)$ is distribution independent and we have our previous formula $C_t = \tau^{-1} C_u$, where $\tau = E(T)$. Note that, in general, C_t is not $\frac{\max I_u(X,Y)}{\max E(T)}$, see [15].

If we have a timing DMC that is also noiseless then we refer to it as a *simple timing channel* (STC). These have been studied in [20]. For a STC, $I_u(X, Y)$ is simply $H(X)$ so

$$C_t = \max \frac{H(X)}{E(T)} \ .$$

(For timing channels C_u is not a useful concept and C_t is understood.) Even for this very trivial type of timing channel, an exact calculation of capacity is difficult. The problem is analogous to finding roots of a polynomial — an easy task if you do it numerically but a very difficult task if you require closed form solutions for the roots [16, 20]. In general, for timing and mixed channels, the capacity analysis is quite difficult.

Capacity Yes, Bandwidth No

Now let us leave the arena of covert channels and just look at one very complicated but important type of communication channel. The reason for this is to drive home the point that we should not use the term bandwidth or maximum bandwidth for capacity. Our communication channel is still memoryless but it is no longer discrete. The input is a continuous signal $f(t)$ with *bandwidth* W. By this we mean that the Fourier Transform $F(\omega)$ of $f(t)$ is zero for $\omega > W$. Furthermore, the signal has an average power P. The output of this channel is the sum of the input signal with independent white noise of power N. Shannon showed [22] that

$$C_t = W \log \left(1 + \frac{P}{N}\right) \ .$$

Therefore, we see that it is wrong to refer to bandwidth or maximum bandwidth as capacity (e.g., [19], [6]) because the bandwidth is a separate characteristic of a continuous channel and the capacity is in fact a function of the bandwidth! We should not reinvent the wheel and use the standard terminology that already exists.

4 The Small Message Criterion

Let us consider discrete communication channels that are noiseless but not necessarily memoryless. Hence a symbol might take a longer amount of time to be transmitted in the future than at other times, or certain symbols may not be transmitted after other symbols (run-length limited and/or time varying channels). Since the channel is noiseless the symbol transmitted is the symbol that is received. Each symbol that is transmitted takes an amount of time to be sent. For a particular time value n, let S_n represent all of the allowed messages that take time n to be transmitted. If the channel is memoryless then S_n consists of all possible symbol sequences whose total transmission time is n, see [20]. If the channel has memory then we can only use the symbol sequences allowed by the parameters of the communication channel. Let $|S_n|$ be the magnitude of the set S_n. Shannon's original definition of capacity for such a channel (he used the ordinary limit) is given by

$$C_t = \limsup_{n \to \infty} \frac{\log |S_n|}{n} \ . \qquad (2)$$

Note for a STC the above Eq. (2), by [20, Thm. 2], reduces to Eq. (1).

Zero Capacity Example
Assume we have a noiseless communication channel with two symbols. The first time the channel is used either symbol can be sent in 1 tick, the second time the channel is used either symbol can be sent in 2 ticks, the third time either symbol can be sent in 4 ticks, ... , the nth time either symbol can be sent in 2^{n-1} ticks.

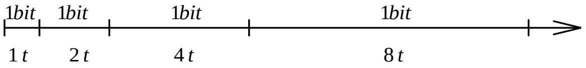

Figure 3

We see that by the nth transmission there are 2^n different messages and that the total transmission time by the nth transmission is $1 + 2 + \cdots + 2^{n-1}$ ticks. Therefore (after some analysis) the capacity of this channel is

$$C_t = \lim_{n \to \infty} \frac{\log 2^n}{\sum_{i=0}^{n-1} 2^i} = \lim_{n \to \infty} \frac{n}{2^n - 1} = 0 \ .$$

However, we can send any message we want with absolutely no loss of fidelity across this channel! Of course as the number of bits that we wish to transmit grows polynomially, the transmission time grows exponentially (this is what the capacity tells us). However, if we have a small message, then who cares what the capacity is? In this example we can noiselessly send a 4 bit message in 15 ticks. Knowing that the capacity is zero does not tell us that we are in a secure situation.

The lesson learned from the above example is an important one and its ideas have been discussed before [17]. If one has a very sensitive but short message then the capacity is not a sufficient measure of security. Also we have previously discussed examples such as these with Wittbold [24].

There are many other examples of this type. Say we have a channel that noiselessly transmits 100 bits in the first tick and then is total noise. The capacity of this channel is zero but the problem is obvious. We need to develop a criterion, or criteria, in addition to capacity, that should be required of secure systems. Capacity, since it is an asymptotic definition, is fine if we are concerned with sending very long files over a long period of time. Then capacity gives us the existence of a (possibly very complicated and long) code that will send messages at a certain level of fidelity at a certain rate.

If our message is short then we need to know what level of covert transmission will be tolerated. Three factors should be taken into account. The first is the length of the message — how many bits? The second factor is the fidelity of the message — is there a threshold below which the degraded short message is no longer a security threat? The third factor is the time frame — if our concern is of a 10 bit message do we care if it takes a second or a week for that message to be transmitted? These three factors make up part of what we envision as a *small message criterion* (SMC). The SMC should reflect the following viewpoint and depends on the triple (n, τ, ρ) :
When a covert channel exists in a system, the SMC will give guidelines for what will be tolerated in terms of covertly leaking a short covert message of length n bits in time τ with fidelity of transmission $\rho\%$. The SMC must be used in conjunction with capacity for a full security analysis/validation of a system.

The SMC can itself be dynamic. If the sensitivity of the messages goes up, the SMC can be tightened up so we can have trade-offs between security and performance. In other words, distinctions should be made between High and Very High, or Critical [2].

We note that previous formal models' work have attempted to capture ideas similar to ours. The various information flow models such as FM and AFM have concerned themselves with how Low probabilities are independent/dependent of High [14, 9]. A particularly interesting formal model has been put forth by Browne in his *Zero Information Finite Sample Theorem* [3]. His theorem captures the information theoretic essence of sending a message for a limited amount of time. We believe that designing a system that satisfies a model is at least as important as building the model itself. In future work we plan to give system designers actual trade-offs between performance and security concerns.

5 Tradeoffs

A communication theorist is concerned with how to send as many bits as possible through a communication channel. A secure system designer, on the other hand, wants to maximize security without ending up with a secure brick. We must examine how security impacts upon performance.

The needs for lowering capacity must be balanced against the needs of performance. As we have discussed, the SMC presents us yet another obstacle to performance, in that lowering capacity alone does not eliminate the threat of covert channels.

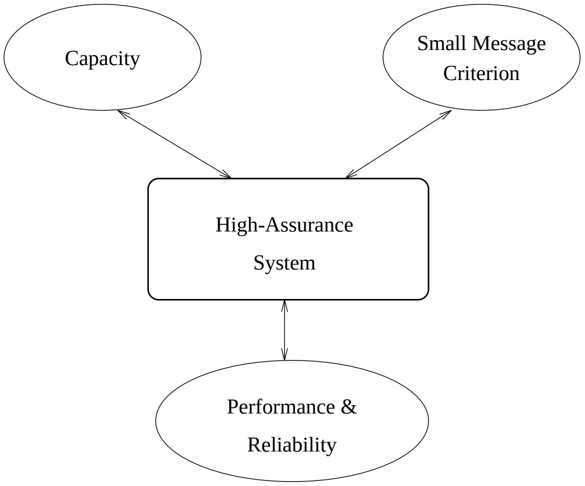

Figure 4: Covert Channel Concerns

We have seen how the need for assurance, via acknowledgements in a DBS, leads to a timing channel. In section 5.1 we will examine another instance of how assurance, via an acknowledgement, leads to a storage DMC in a DBS. We will then quantify how methods of lowering covert channel capacity lead to a decline in performance. We will not go into specific instances of how the SMC can affect performance.

Let us examine Eq. (1) more carefully. We may express C_t, for a DMC, as

$$C_t = \max \frac{H(X) - H(X \mid Y)}{E(T)} \ .$$

We see that there are two, not necessarily independent, ways of lowering capacity. We can either try to increase the times that it takes for symbols to pass over the channel—thus decreasing $1/E(T)$, or, we can increase the noise, which will increase $H(X \mid Y)$, which will in turn decrease $I_u(X, Y)$. If we did not care about performance this would be wonderful. Unfortunately, we do care about performance.

In section 5.1 we discuss an approach to increasing noise in a storage channel, which has the unfortunate effect of lowering performance. In section 5.2 we discuss our recent work on a way of lowering capacity and meeting the SMC, for a timing channel, without sacrificing performance by using the "pump".

5.1 Two Phase Commit Protocol

A multilevel secure replicated architecture database system (MLS-RA DBS) using the two phase commit protocol (2PC) for atomic commitment results in a storage channel. This is no surprise and its mathematical details have been studied in [5]. We will briefly summarize a simple idealized version of it here.

In a MLS-RA DBS, copies of lower data are retained in replicated higher copies. When a particular low user (we will use the term user for any user or process) designated as Low, wishes to update a data item all of the higher copies have to agree to the update via a commit, if even one of them aborts then the low data item is not updated and the Low user receives an abort. All of the abort/commit voting is moderated through a trusted front end. This prevents Low from knowing how the other users voted. However, if there is one particular high user, designated as High, that wishes to communicate covertly with Low, it may do so through this scheme. Assume there are $(n+2)$ users; Low, High, and n other users with levels higher than Low. A Trojan horse is in place so that Low and High can always commit or abort as they wish. The covert communication works as follows:

Low wishes to update the DBS. If Low receives a commit then it interprets that as a 0; if Low receives an abort then it interprets that as a 1. High can send a 1 to Low with no noise simply by voting to abort. Therefore $P(Low = 1 \mid High = 1) = 1$. However, if High wishes to send a 0 to Low by voting to commit then there is the possibility that one of the other users might vote to abort. Hence, the transmission of the 0 is noisy. We assign the probability p to another user aborting, and assume that everything takes one tick, where a tick is the standard unit of time, so $C_u = C_t$. (The actual question of delaying the votes is more complicated and will not be looked at here.)

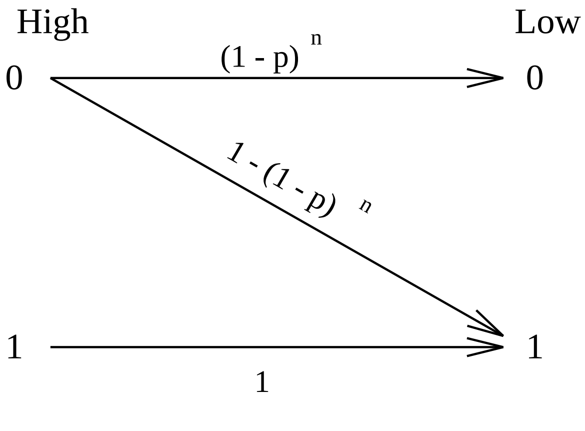

Figure 5: Z-Channel

This set-up forms what is known as a Z-channel [8]. $P(Low = 0 \mid High = 0) = (1-p)^n$, since all of the other n users must vote to commit; High sending a 0 and Low getting a 1 comes about as the complement of all of the other users voting to commit, thus $P(low = 1 \mid High = 0) = 1 - (1-p)^n$. As a security counter-measure, the system itself could increase p (increase the noise) to lower capacity. In what follows we show that this has the deleterious side effect of lowering performance.

If there is no Trojan horse present then we want to examine, as a measure of performance, how long Low has to wait, on the average, for a global commit. Since there are no longer any Trojan horses present, all $(n+2)$ users abort with probability p. Let τ be the random variable representing how many times Low must try to commit an update. $P(\tau = k)$, k a positive integer representing the number of ticks, is given by

$$P(\tau = k) = \left(1 - (1-p)^{n+2}\right)^{k-1}(1-p)^{n+2}$$

Therefore τ is a geometric random variable and its mean is $E(\tau) = \frac{1}{(1-p)^{n+2}}$. Thus as p increases so does $E(\tau)$ (hence lowering performance), however p increasing causes C_t to decrease (hence increasing security). Hence, we see that there is a trade-off, as the two following plots demonstrate.

Figure 6 is a plot of $E(\tau)$ for $n = 13$. Figure 7 is a dimensionless plot of C_t and $1/E(\tau)$ for $n = 13$, and it shows the striking relationship of diminishing capacity to diminishing performance. The calculations for capacity can be found in [8] and [5].

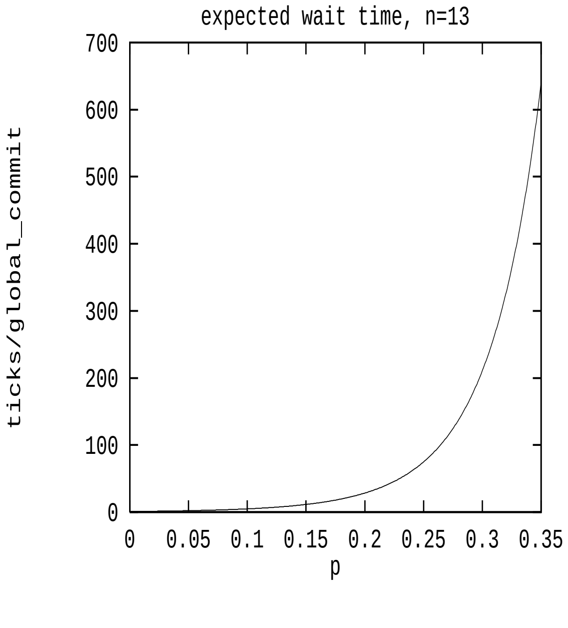

Figure 6: mean wait, n=13

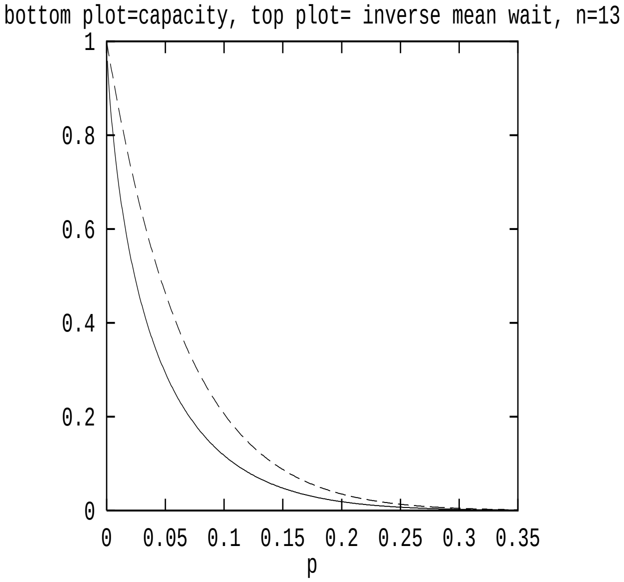

Figure 7: lowering capacity lowers performance

We will not address the SMC for this 2PC covert storage channel.

5.2 The "pump"

The purpose of the pump is to provide a reliable, recoverable, and quasi-secure low-to-high communication scheme. We summarize from an earlier paper of ours [11]. The pump is designed to be quasi-secure because it does not sacrifice performance when Low and High response times are approximately equal. The pump works as follows:

- A low process (Low) sends a message to the pump with the address of its (high) destination. If Low receives an *ack* then Low assumes that the message will be safely delivered to its destination and sends the next message.
- When the pump receives a message from Low, it stores the message in its buffer and sends an *ack* to Low.
- The pump delivers messages that are stored in its buffer to the proper destinations. When there is an *ack* from the destination, the pump knows that the message has been safely delivered to its destination and deletes the message from its buffer.

Since the timing of *ack*s to Low can be used as a covert timing channel, the pump introduces random noise when it sends an *ack* to Low. This random noise is a function of a chosen probability distribution and the moving average of *ack* times from destinations to the pump.

The pump does not sacrifice performance to lower capacity. Another interesting aspect of the pump is that it is sensitive to the SMC. We have shown [11, Sec. 5.1] how the construction of the pump makes it very difficult to send even the smallest of messages in a small amount of time. The use of the distribution parameters gives the system designer control parameters that can be adjusted to meet different security criteria. We feel that this is a fruitful avenue for future work.

6 Conclusions

In this paper we presented our position that covert channels can never be totally eliminated from high-assurance computing systems. We discussed two major misconceptions in the theory of covert channels. The first is the use of the term bandwidth and the second is the reliance on capacity alone as a measure of covert channel vulnerability. We then introduced the small message criterion as a way of supplementing capacity to evaluate the covert channel threat.

Finally, we discussed ways, e.g., the pump, of minimizing the threat of covert channels without drastically reducing performance.

References

[1] D.E. Bell and L.J. La Padula. *Secure Computer System: Unified Exposition and Multics Interpretation, MTR-2997.* MITRE Corp., Bedford, MA, March 1976.

[2] Peter K. Boucher, Raymond K. Clark, Ira B. Greenberg, E. Douglas Jensen, and Douglas M. Wells. Toward a multilevel-secure, best-effort real-time scheduler. In *(Preprints of the) 4th International Working Conference on Dependable Computing for Critical Applications*, pages 33–45, San Diego, CA, January 1994.

[3] Randy Browne. The Turing test and non-information flow. In *Proc. of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 373–385, Oakland, CA, May 1991.

[4] Oliver Costich and Sushil Jajodia. Maintaining transaction atomicity in multilevel secure database systems with kernelized architecture. *Database Security VI: Status and Prospects*, pages 249–265, 1993.

[5] Oliver L. Costich and Ira S. Moskowitz. Analysis of a storage channel in the two-phase commit protocol. In *Proc. of The Computer Security Foundations Workshop 4*, pages 201–208, Franconia, NH, June 1991.

[6] Department of Defense, National Computer Security Center. *Trusted Computer System Evaluation Criteria 5200.28-STD*, December 1985.

[7] T. Lunt et el. The seaview security model. *IEEE Transaction on Software Engineering*, 16(6):593–607, 1990.

[8] Solomon W. Golomb. The limiting behavior of the Z-channel. *IEEE Transactions on Information Theory*, page 372, May 1980.

[9] James W. Gray, III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3,4):255–294, 1992.

[10] Sushil Jajodia and Boris Kogan. Integrating an object-oriented data model with multilevel security. In *Proc. of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 76–85, Oakland, CA, 1990.

[11] Myong H. Kang and Ira S. Moskowitz. A pump for rapid, reliable, secure communication. In *Proc. of the 1st ACM Conference on Computer and Communications Security*, pages 119–129, Fairfax, VA, November 1993.

[12] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10), October 1973.

[13] Amit G. Mathur and Thomas F. Keefe. The concurrency control and recovery problem for multilevel update transactions in MLS systems. In *Proc. of the Workshop on Computer Security Foundations VI*, pages 10–23, Franconia, NH, June 1993.

[14] John McLean. Security models and information flow. In *Proc. of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 180–187, Oakland, CA, May 1990.

[15] Jonathan K. Millen. Finite-state noiseless covert channels. In *Proc. of The Computer Security Foundations Workshop II*, pages 81–86, Franconia, NH, June 1989.

[16] Allen R. Miller and Ira S. Moskowitz. Reduction of a class of Fox-Wright Psi functions for certain rational parameters. 1993 Preprint.

[17] Robert Morris. Private Communication and comments at various meetings.

[18] Ira S. Moskowitz and Allen R. Miller. The channel capacity of a certain noisy timing channel. *IEEE Transactions on Information Theory*, 38(4):1339–1344, July 1992.

[19] Ira S. Moskowitz and Allen R. Miller. The influence of delay upon an idealized channel's bandwidth. In *Proc. of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, May 1992.

[20] Ira S. Moskowitz and Allen R. Miller. Simple timing channels. In *Proc. of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, May 1994.

[21] Ravi S. Sandhu, Roshan Thomas, and Sushil Jodia. Supporting timing-channel free computations in multilevel secure object-oriented databases. *Database Security, V:Status and Prospects*, pages 297–314, 1992.

[22] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, IL, 1949. Also appeared as a series of papers by Shannon in the Bell System Technical Journal, July 1948, October 1948 (A Mathematical Theory of Communication), January 1949 (Communication in the Presence of Noise).

[23] Sergio Verdú. On channel capacity per unit cost. *IEEE Transactions on Information Theory*, 36(5):1019–1030, September 1990.

[24] Todd Wittbold. Private Communication.